# jQuery

## Basics

It is a JavaScript library specialized for changing web page documents on the fly.

jQuery uses selector engine.

|  |  |
| --- | --- |
| **JavaScript** | **jQuery** |
| document.getElementByTagName(“p”)  [0].innerHTML = “Change”; | $(“p”).html(“Change”) |
| function init(){} window.onload = init; | $(document).ready(function(){}); |

The JavaScript interpreter doesn’t change the original HTML and CSS files. It makes changes to the DOM’s representation of the page in the browser’s memory.

The dollar sign “$” with the parentheses is the shorter name of the jQuery function. This shortcut saves us from writing jQuery(). The jQuery function is also often referred to as the **jQuery wrapper**.

jQuery() 🡪 $()

$() can have css selector, JavaScript Object, HTML

If we put a **CSS selector** here, jQuery will return us the **set of elements** that match that selector

If we put a string of **HTML** in here, we can add DOM elements to the browser page.

**JavaScript Object** can also be used as a selector.

### CSS Selector

CSS selectors select elements to add style to those elements; jQuery selectors select elements to add behavior to those elements.

### Element Selector

h1{

text-align: left;

}

### Class Selector

.my\_class{

position: absolute;

}

### ID Selector

#my\_id{

color: #3300FF

}

### jQuery methods

.hide() 🡪 Hides the element

.show() 🡪 Shows the element

.toggle() 🡪 Hide and Show the element

.slideUp() 🡪 It changes the height property of the element until its 0

.slideDown() 🡪 It changes the height property of the element from 0

.slideToggle() 🡪 Slides up and down the element

.fadeIn() 🡪 Fades in the element

.fadeOut() 🡪 Fades out the element

.fadeTo() 🡪 Fades to the element

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> jQuery </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "basics.js"></**script**>

</**head**>

<**body**>

<**p** id = "text"> Hide and Show </**p**>

<**button** id = "hide">Hide</**button**>

<**button** id = "show">Show</**button**>

<**button** id = "toggle">Toggle</**button**><**br**/><**br**/>

<**button** id = "slide-up">Slide Up</**button**>

<**button** id = "slide-down">Slide Down</**button**>

<**button** id = "slide-toggle">Slide Toggle</**button**><**br**/><**br**/>

<**button** id = "fade-in">Fade In</**button**>

<**button** id = "fade-out">Fade Out</**button**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#hide").click(**function**(){

$("#text").hide(2000);

});

$("#show").click(**function**(){

$("#text").show(2000);

});

$("#toggle").click(**function**(){

$("#text").toggle(2000);

});

$("#slide-up").click(**function**(){

$("#text").slideUp(2000);

});

$("#slide-down").click(**function**(){

$("#text").slideDown(2000);

});

$("#slide-toggle").click(**function**(){

$("#text").slideToggle(2000);

});

$("#fade-in").click(**function**(){

$("#text").fadeIn(2000);

});

$("#fade-out").click(**function**(){

$("#text").fadeOut(2000);

});

});

**Code 1: basics.js**

**$(this)**, this selector gives us an easy way to point to the current element. It’s important to think $(this) as **context-dependent.**

### Descendant Selectors

html

↑

body

↑

div

↑

div div p id = “good”

↑ ↑

img p

$(“div p#good”)

$(“div div”)

$(“div p”)

$(“div div img”)

## Binding an Event

* **$(“#myElement”).click(function(){})**
* **$(“#myElement”).bind(“click”,function(){})**

The first method is simply a shortcut for second method, but only **when the DOM elements exist already**

### Events

**Selector + Event + Function = Complex Interaction**

|  |  |
| --- | --- |
| **Events** | **Description** |
| click | Mouse Events 🡪 Click |
| dblclick | Mouse Events 🡪 Double Click |
| focusin | Mouse Events 🡪 Focus In |
| focusout | Mouse Events 🡪 Focus Out |
| hover | Mouse Events 🡪 Hover |
| mousedown | Mouse Events 🡪 Mouse Down |
| mouseenter | Mouse Events 🡪 Mouse Enter |
| mouseleave | Mouse Events 🡪 Mouse Leave |
| mousemove | Mouse Events 🡪 Mouse Move |
| mouseout | Mouse Events 🡪 Mouse Out |
| mouseover | Mouse Events 🡪 Mouse Over |
| mouseup | Mouse Events 🡪 Mouse Up |
| toggle | Mouse Events 🡪 Toggle |
| keydown | Keyboard Events 🡪 Press Down Arrow |
| keypress | Keyboard Events 🡪 Press Enter |
| keypup | Keyboard Events 🡪 Press Up Arrow |
| blur | Form Events 🡪 Focus out of the field box |
| change | Form Events 🡪 Change in the field Text |
| focus | Form Events 🡪 Focusing in the field box |
| select | Form Events 🡪 Selecting the field box |
| submit | Form Events 🡪 Submit Clicked |
| load | Document Events 🡪 Page Fully Loaded |
| ready | Document Events 🡪 Page is ready |
| unload | Document Events 🡪 Page unloaded |
| error | Browser Events 🡪 Error in Browser |
| resize | Browser Events 🡪 Resize the Browser |
| scroll | Browser Events 🡪 Scroll through the Browser |

### Removing Events

$(“#myElement”).unbind(“click”) 🡪 Removes the click event from myElement

$(“#myElement”).unbind() 🡪 Removes all the events from myElement

### Going through Elements

$(“.nav-items”).each(function(){});

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Selectors</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "selectors.js"></**script**>

</**head**>

<**body**>

<**div** id = "image-container">

<**img** src = "images/desktop.ico" class = "picture"/>

<**img** src = "images/hdd.ico" class = "picture"/>

<**img** src = "images/trash.ico" class = "picture"/>

<**img** src = "images/folder.ico" class = "picture"/>

</**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$(".picture").click(**function**(){

$(**this**).slideUp();

**var** discount = Math.floor((Math.random()\*5 + 5));

**var** message = "<p> Your discount price is: " + discount + "%</p>";

$("#image-container").append(message);

$(".picture").each(**function**(){

$(**this**).unbind("click");

});

$(".picture").hide(5000);

});

});

**Code 2: selectors.js**

### Checking Element

**$.contains(document.getElementById(“parent”), document.getElementById(“child”))**

This method check if the first parameter (parent) contains the second parameter(child)

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Check</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "check.js"></**script**>

</**head**>

<**body**>

<**div** id = "container">

<**button** id = "check"> Check </**button**>

</**div**>

<**button** id = "outsider"> Outsider</**button**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#check").click(**function**(){

console.log($.contains(document.getElementById("container"), **this**));

**var** outsider = document.getElementById("outsider");

console.log($.contains(document.getElementById("container"), outsider));

});

});

**Code 3: check.js**

### CSS Hover

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Check</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "hover.js"></**script**>

</**head**>

<**body**>

<**div** id = "container">

<**button** id = "hover"> Hover </**button**>

</**div**>

</**body**>

<**style**>

#hover{

**border**: **none**;

**padding**: 5px;

transition: 0.5s;

}

**.hover-on**{

**color**: rgba(140, 234, 12, 0.5);

**background-color**: rgba(0, 0, 0, 0.6);

transition: 0.5s;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$("#hover").bind("mouseover",**function**(){

**if**($(**this**).hasClass("hover-on")){}

**else**{$(**this**).addClass("hover-on");}

});

$("#hover").bind("mouseleave", **function**(){

**if**($(**this**).hasClass("hover-on")){

$(**this**).removeClass("hover-on");

}

});

});

**Code 4: hover.js**

## Modify the Document

**$(“img#thumbnail”).remove()**

This removes the method drops the element out of the DOM

**$(“img#thumbnail”).detach()**

This detach method takes the selected elements(s) but holds on to it so that it can be reattached later.

**$(“.fish”).parent()**

It selects all the elements in the fish class and then get the element above those element

**$(“.menu\_list”).children()**

It selects all the elements in the menu\_list get all the elements below it

**$(“.fish”).prev()**

Selects all the element of fish class and get the immediate left sibling

**$(“.fish”).next()**

Selects all the element of fish class and get the immediate right sibling

**$(“h3”).replaceWith(“<h1>My Menu</h1>”);**

Selects all the h3 elements, replace with selected elements with the contents in parenthesis.

It works well when we have a one-to-one replacement.

**$(“.meat”).before(“<li>Tofu</li>”);**

before inserts content before the selected element

**$(“.meat”).after(“<li>Tofu</li>”);**

after inserts content after the selected element

**$(“.menu\_list”).children().first();**

The first method will filter out everything but the first element in a selected set of elements

**$(“.menu\_list”).children().last();**

The last method will filter out everything bu the last element in a selected set of elements.

**$(“.menu\_list”).children().eq(0);**

The **eq** method will filter out everything out the element whose index number is equal to what we put in the parenthesis in a selected set of elements.

**$(“.menu\_list”).children().slice(1,3);**

The slice method will filter out everything but elements with an index between the index number we put it parenthesis.

**$(“.menu\_list”).children().filter(“.organic”);**

The filter method will filter out everything but elements that match the selector we put in its parenthesesis

**$(“.menu\_list”).children().not(“.local”);**

The not method will filter out everything that does not match the selector we place in the parentheses.

### Storing the Variable

**$f = $(“.fish”).parent().parent().detach();**

jQuery variable

**$f[2] = 15;**

jQuery array, **“$”** it is just a coding convention

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Set Menu</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "food.js"></**script**>

</**head**>

<**body**>

<**button** id = "vegetarian">Vegetarian</**button**>

<**button** id = "restore">Restore</**button**>

<**div** id = "menu-warapper">

<**h4**>Dinner Entrees</**h4**>

<**ul** class = "menu-entrees">

<**li**>Thai-Style Halibut

<**ul** class = "menu-list">

<**li**>coconut milk</**li**>

<**li**>pan-fried halibut</**li**>

<**li**>early autumn vegetables</**li**>

<**li**>thai spices</**li**>

</**ul**>

</**li**>

<**li**>House Grilled Panini

<**ul** class = "menu-list">

<**li** class = "meat">prosciutto</**li**>

<**li**>provolone</**li**>

<**li**>avocado</**li**>

<**li**>sourdough</**li**>

</**ul**>

</**li**>

<**li**>Southwest Slider

<**ul** class = "menu-list">

<**li**>whole chiles</**li**>

<**li** class = "meat">hamburger</**li**>

<**li**>pepperjack cheese</**li**>

<**li**>multigrain roll</**li**>

</**ul**>

</**li**>

</**ul**>

</**div**>

</**body**>

</**html**>

**var** $deleted = [];

$(document).ready(**function**(){

$("#vegetarian").click(**function**(){

$deleted.push($(".menu-list > .meat")[0]);

$deleted.push($(".menu-list > .meat")[1]);

$(".menu-list > .meat").each(**function**(){

$(**this**).detach();

})

});

$("#restore").click(**function**(){

$(".menu-list").children().eq(4).before($deleted[0]);

$(".menu-list").children().eq(9).before($deleted[1]);

});

});

**Code 5: food.js**

## Effects and Animation

**Lightning Animation**

**$("#lightning").fadeOut(Math.floor(Math.random()\*2000))**

**.fadeIn(Math.floor(Math.random()\*2000));**

This animation will cause lightning effect

**Animation**

The **.animate()** will only work on CSS properties that use numbers for their settings.

|  |  |
| --- | --- |
| borders, margin, padding | bottom, left, right, and top position |
| element height, min-height, and maz-height | background position |
| element width, min-width, and max-width | letter spacing, word spacing |
| font size | text indent, line height |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Animation</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "animation.js"></**script**>

</**head**>

<**body**>

<**img** src = "images/lightning.png" alt = "light1" id = "light1"/>

<**img** src = "images/lightning2.png" alt = "light2" id = "light2"/><**br**/><**br**/>

<**img** src = "images/desktop.ico" id = "desktop" class = "icon"/>

<**img** src = "images/folder.ico" id = "folder" class = "icon"/>

<**img** src = "images/hdd.ico" id = "hdd" class = "icon"/>

<**img** src = "images/trash.ico" id = "trash" class = "icon"/>

</**body**>

<**style**>

#light1, #light2{

**width**: 5%;

**height**: 5%;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$(window).focus(blink);

$(".icon").click(**function**(){

$(**this**).animate({

opacity: 0,

width: "200",

height: "200",

},5000);

});

});

**function** blink(){

$("#light1").fadeOut(Math.floor(Math.random()\*2000))

.fadeIn(Math.floor(Math.random()\*2000));

$("#light2").fadeOut(Math.floor(Math.random()\*2000))

.fadeIn(Math.floor(Math.random()\*2000));

setInterval(blink, 5000);

}

**Code 5: animation.js**

### Slide Left or Right

We need to make sure the animated element needs to have a position property to relative.

**.icon{position: relative};**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Animation</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "animation.js"></**script**>

</**head**>

<**body**>

<**img** src = "images/desktop.ico" id = "desktop" class = "icon"/>

<**img** src = "images/folder.ico" id = "folder" class = "icon"/>

<**img** src = "images/hdd.ico" id = "hdd" class = "icon"/>

<**img** src = "images/trash.ico" id = "trash" class = "icon"/>

</**body**>

<**style**>

**.icon**{

**position**: **relative**;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$(".icon").click(**function**(){

$(**this**).animate({

right: "+=100px"

}, 5000);

});

});

**Code 6: slideleft.js**

**jQuery is great for the elements that already exists in the document**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Attribute</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "attr.js"></**script**>

</**head**>

<**body**>

<**button** id = "showImage">Show Image</**button**>

<**div** id = "image-container"></**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#showImage").click(**function**(){

**if**(!document.getElementById("desktop")){

**var** image = document.createElement("img");

$(image).attr("src", "images/desktop.ico");

$(image).attr("id", "desktop");

$("#image-container").append(image);

$("#showImage").html("Boom I am back!");

}**else**{

**var** image = document.getElementById("desktop");

document.getElementById("image-container").removeChild(image);

$("#showImage").html("Click me again!");

}

});

});

**Code 7: attr.js**

## Window Events

|  |  |
| --- | --- |
| **window.name** | property, let us access or set the name of the window |
| **window.history** | property, let us access the different URLs that the window has loaded over time |
| **window.document** | property, main content of the loaded document |
| **window.onfocus** | property, detects when the window receives a click, keyboard input or any kind of input |
| **window.setTimeout()** | method, set a period of time to wait before calling a function or other statement |
| **window.clearTimeout()** | method, cancel the period of time of wait |
| **window.setInterval()** | method, set a period of time to wait between repetitions of a function call or other statement |
| **window.clearInterval()** | method, cancel the period of time to wait between repetitions |
| **window.onblur()** | detects when the window loses focus |

## Jquery Ajax

Asynchronous JavaScript and eXtensible Markup Language

<?php

$employees = ["sam" => "engineer", "bob" => "housekeeper"];

**if**(isset($\_REQUEST['postit'])):

**echo** json\_encode($employees);

**elseif**(isset($\_REQUEST['getit'])):

**echo** json\_encode($employees);

**else**:

**endif**;

?>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Ajax</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "ajax.js"></**script**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$.ajax({

url: "ajax.php?getit=confirm",

cache: **false**,

dataType: "json",

success: **function**(json\_data){

**var** employees = json\_data;

console.log(employees.sam);

console.log(employees.bob);

}

});

**var** url = "ajax.php?getit=confirm";

$.get(url, **function**(data, status){

console.log(data, status);

});

**var** url = "ajax.php";

**var** data = "postit=confirm"

$.post(url, data, **function**(data, status){

console.log(data, status);

});

**var** url = "ajax.php?getit=confirm";

$.getJSON(url, **function**(employees, status){

console.log(employees.sam);

})

});

**Code 7: ajax.js**

**Javascript Version**

|  |  |
| --- | --- |
| **Post** | **Get** |
| request = new XMLHttpRequest(); | request = new XMLHttpRequest(); |
| var url = **"data.php"**; | var url = "**data.php?call=calling**"; |
| var requestData = **"call=calling"**; | var requestData = **null**; |
| request.open(**"POST"**, url, true); | request.open(**"GET"**, url, true); |
| request.onreadystatechange = func | request.onreadystatechange = func |
| **request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");**  Only Required for Post | |
| request.send(**requestData**); | request.send(**null**)**;** |

**jQuery Version**

|  |  |
| --- | --- |
| **Post** | **Get** |
| $.post(*URL,data,callback*); | $.get(*URL,callback*); |
| $.post(url, {name: “”, age: “”}  function(data, status){}  ); | $.get(url,  function(data,status){}  ); |
| **Works for Both**  $.ajax({  url: “”,  cache: false,  dataType: “json”,  success: function(data){}  }); | |
| **Other Shortcuts**  **$.getJSON()**  **$.getScript()**  **$.load()** | |

## Serialize

<form id = “myform>

<input type = “text” name = “a” value = “1”/>

<input type = “text” name = “b” value = “2”/>

<input type = “submit” name = “submit”/>

**$(“#myform”).serialize()**

The form ID selector serializes the method

a=1&b=2

🡪The value of submit will not show up

**$(“#myform:input”).serializeArray()**

The form id takes only data from the input and then transform into an array

{

{

name: “a”,

value: “1”

}

{

name: “b”,

value: “2”

}

}

🡪The value of submit will not show up

**Advantages of Serialization**

|  |
| --- |
| **Normal Way (Form Data)** |
| //**JavaScript Way**  var a = document.querySelector(“input[name=’a’]);  var value = a.value();  var data = “a=” + value  //**jQuery Way**  var b = $(“input[name=’b’]”).val();  var data = “b=” + b  //**Serialize Way** (Get all the values)  var form\_id\_data = $(“#myform”).serializa() |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Serial </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "serial.js"></**script**>

</**head**>

<**body**>

<**form** id = "myForm">

<**label** for = "fullname"> Full Name </**label**><**br**/>

<**input** type = "text" name = "fullname"><**br**/><**br**/>

<**label** for = "age"> Age </**label**><**br**/>

<**input** type = "text" name = "age"><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm">

</**form**>

</**body**>

</**html**>

<?php

**if**(isset($\_POST)):

$fullname = $\_POST['fullname'];

$age = $\_POST['age'];

$fullname = "FullName: ".$fullname;

$age = "Age: ".$age;

**try**{

$file = fopen("serial.txt", "w");

$string = $fullname.", ".$age;

fwrite($file, $string);

var\_dump($\_POST);

}**catch**(Exception $e){

**echo** ("Error: ".$e->getMessage());

}**finally**{

fclose($file);

}

**endif**;

?>

$(document).ready(**function**(){

$("#myForm").submit(**function**(e){

e.preventDefault();

*// var data = $("#myForm:input").serializeArray();*

**var** data = $("#myForm").serialize();

**var** url = "serial.php";

$.ajax({

url: url,

type: 'post',

data: data,

success: **function**(data){

alert(data);

}

});

});

});

**Code 8: serialize.js**

## jQueryUI

jQuery offers a plug-in architecture that allows web developers to extend (or add onto) the core jQuery library.

**Effects plug-ins**

It extends jQuery by adding more effects. Make our elements bounce, explode, pulsate, or shake. jQuery UIs also includes easing functions, complex mathematical operations that make animations look more realistic.

**Interaction plug-ins**

Interactions add more complex behavior to web apps. We can enable user to interact with elements by making those elements draggable, droppable or sortable.

**Widget plug-ins**

A web widget is a self-contained component that adds functionality to our web app. Widgets save our tons of coding time and complexity while creating useable and responsive user interface elements.

**Plugins**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Puff | Effect | Makes an element appear to expand and dissipate into transparency like smoke |
| Autocomplete | Widget | Provides a list of possible values when a user types into an input field |
| Droppable | Interaction | Makes a DOM element a target for draggable elements |
| Explode | Effect | Makes an element appear to break into pieces and spread out in several directions. |
| Sortable | Interaction | Makes an element sortable by dragging |
| Progressbar | Widget | Displays the current percentage of completion for some event |
| Resizable | Interaction | Gives an element sortable by dragging |
| Blind | Effect | Makes an element appear to slide up or down like a window treatment |
| Accordion | Widget | Creates stacked and collapsible areas to organize web content |

### Date Picker

$(“#datepicker”)**.datepicker()**

.datepicker({

stepMonths: 3,

changeMonth: true

})

Options:

**stepMonths** 🡪Three months step backward or forward

**changeMonth** 🡪User can choose the month from a drop-down list

**changeYear** 🡪 User can change the year

### Button/Button Set

$("#radio")**.buttonset();**

$("input[type='radio']")**.button();**

### Slider

$("#slider")**.slider();**

$("#slider").slider({

value: 0,

min: 0,

max: 100,

step: 10,

orientation: "horizontal",

slide: function(event, ui){

$("#slider\_value").val(ui.value);

}

});

Options

**value** 🡪 It tells the slider what value to start with

**min** 🡪 It tells the slider the lowest value a user can enter

**max** 🡪 It tells the slider the highest value a user can enter

**step 🡪** It tells the slider what increments we want the values in

**orientation 🡪** It can be horizontal or vertical

**slide 🡪** It is the slide event handler. The user triggers the slide event when the slide is moved. The slide event is attached to a function callback. When the function runs, this sets the input with the jQuery val method.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> jQuery UI</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "required/external/jquery/jquery.js"></**script**>

<**link** rel = "stylesheet" href = "required/jquery-ui.css"/>

<**script** src = "required/jquery-ui.js"></**script**>

<**script** src = "ui.js"></**script**>

</**head**>

<**body**>

<**form** id = "myForm">

<**label** for = "datepicker"> Date </**label**><**br**/>

<**input** type = "text" name = "datepicker" id = "datepicker"/><**br**/><**br**/>

<**div** id = "radio">

<**label** for = "gender"> Gender </**label**><**br**/>

<**input** type = "radio" name = "gender" id = "gender1" value = "male">

<**label** for = "gender1"> Male </**label**>

<**input** type = "radio" name = "gender" id = "gender2" value = "female">

<**label** for = "gender2"> Female </**label**>

</**div**><**br**/><**br**/>

<**input** type = "text" id = "slider\_value" readonly = "readonly"><**br**/>

<**div** id = "slider"></**div**><**br**/><**br**/>

<**canvas** id = "display" height = "100px" width = "100px"></**canvas**><**br**/>

<**input** type = "text" id = "display\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "red"></**div**>

<**input** type = "text" id = "red\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "green"></**div**>

<**input** type = "text" id = "green\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "blue"></**div**>

<**input** type = "text" id = "blue\_value" readonly = "readonly"><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm"/>

</**form**>

<**style**>

#myForm{

**width**: 15%;

**text-align**: **left**;

}

</**style**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#datepicker").datepicker({

changeMonth: **true**,

changeYear: **true**

});

$("#radio").buttonset();

*// $("input[type='radio']").button();*

$("#slider").slider({

value: 0,

min: 0,

max: 100,

step: 10,

orientation: "horizontal",

slide: **function**(event, ui){

$("#slider\_value").val(ui.value);

}

});

$("#red, #green, #blue").slider({

value: 127,

min: 0,

max: 255,

orientation: "horizontal",

slide: refreshDisplay,

change: refreshDisplay

});

**function** refreshDisplay(){

**var** red\_value = $("#red").slider("value");

**var** green\_value = $("#green").slider("value");

**var** blue\_value = $("#blue").slider("value");

$("#red\_value").val(red\_value);

$("#green\_value").val(green\_value);

$("#blue\_value").val(blue\_value);

**var** string = red\_value + ', ' + green\_value + ", " + blue\_value;

$("#display\_value").val(string);

**var** display = document.getElementById("display");

**var** image = display.getContext('2d');

image.fillRect(0, 0, 50, 50);

image.fillStyle = "rgba("+ red\_value + "," +

green\_value + "," +

blue\_value + ")";

}

});

**Code 8: ui.js**

## CDNs

Content delivery networks, or content distribution networks) are large networks of servers, designed to store and deliver information—data, software, API code, media files or videos, etc. – making it easily accessible on the Web. Windows Azure and Amazon CloudFront are examples of traditional CDNs.

## Namespace

**$.noConflict();**

**jQuery(document.ready(function($){});**

We will be only need to use this if we plan to use other JavaScript librarires that use $ as a reference.

## Queues

Queues in jQuery are mostly used for animations. There is an array of functions stored on a per-element basic, using jQuery.data. There are firs-in-first=out(FIFO). We can add a function to the queue by calling **.queue and we remove (by calling) the function using .dequeue**

Every element can have one to many queues of functions attached to it by jQuery. In most applications, only one queue (called **fx**) is used. queues allow a sequence of actions to be called on an element asynchronously., without halting program execution.